SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR 2

GENERATIVNE SUPARNIČKE MREŽE

Tomislav Krog

Sadržaj

Uvod 1

1. Generativne suparničke mreže 2

1.1. Općenito o GAN-u 2

1.2. Generator model 3

1.3. Diskriminator model 4

1.4. Arhitektura GAN-a 5

2. Implementacija i rezultati 8

2.1. Skup podataka 8

2.2. Programska implementacija 8

2.3. Rezultati 10

Zaključak 14

Literatura 15

Sažetak 16

Summary 17

Skraćenice 18

Uvod

Duboke neuronske mreže postale su popularan pristup u strojnom učenju za obradu različitih vrsta podataka, kao što su slike, zvuk i prirodni jezik. Međutim, često se javlja potreba za generiranjem novih primjera podataka koji su slični onima koji su korišteni za obuku mreže. U takvim slučajevima, generativne suparničke mreže (GAN-ovi) pokazale su se kao učinkovit alat za generiranje novih primjera podataka.

GAN-ovi se sastoje od dva neuronska modela koji rade zajedno - generatora i diskriminatora. Generator stvara nove primjere podataka koji su slični onima koji su korišteni za obuku mreže, dok diskriminator procjenjuje njihovu autentičnost u odnosu na primjere iz stvarnog skupa podataka. Cilj je optimizirati modele tako da generator može stvarati podatke koji su dovoljno uvjerljivi da diskriminator ne može razlikovati lažne primjere od stvarnih.

U ovom radu, istražujemo primjenu GAN-ova na skupu podataka MNIST koji se sastoji od rukom pisanih znamenki. Cilj je generirati nove primjere rukom pisanih znamenki koje su vizualno slične onima iz MNIST skupa podataka.

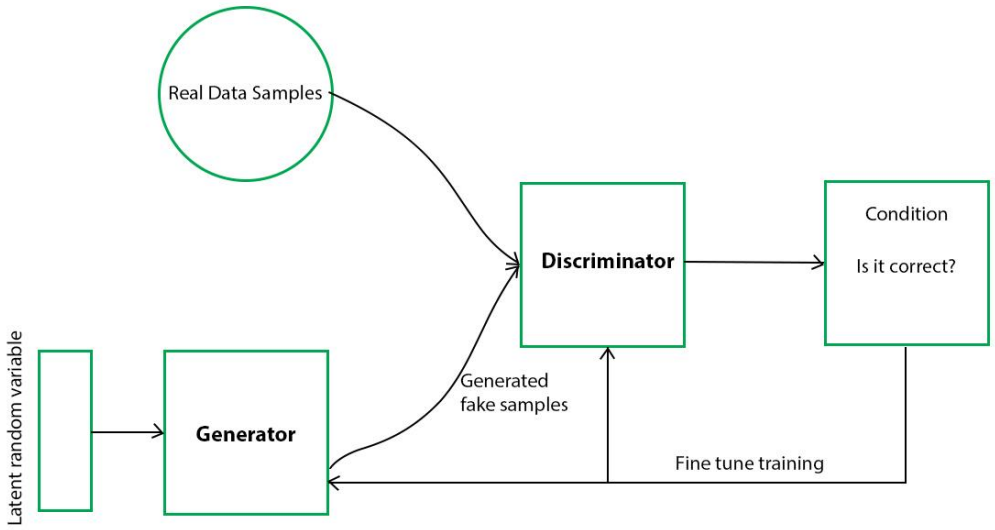
1. Generativne suparničke mreže

1.1. Općenito o GAN-u

Generativne suparničke mreže (GAN) pristup su generativnom modeliranju koji koristi duboko učenje, poput konvolucijskih neuronskih mreža.

Generativno modeliranje je zadatak nenadziranog učenja u strojnom učenju koji uključuje automatsko otkrivanje i učenje pravilnosti ili uzoraka u ulaznim podacima na način da model može generirati nove primjere koji bi vjerojatno mogli biti izvučeni iz izvornog skupa podataka.

GAN-ovi su pametan način treniranja generativnog modela koji problem postavlja kao problem nadziranog učenja s dva podsustava: model generatora koji treniramo da generira nove primjere, i model diskriminatora koji pokušava klasificirati primjere kao stvarne (iz domene) ili lažne (generirane). Dva se modela zajedno treniraju u igri nulte sume, odnosno suprotstavljenosti, dok diskriminator model ne bude zavaravan oko polovice vremena, što znači da generator model generira uvjerljive primjere [1]. Na slici 1.1 prikazana je generalna shema GAN-a.



Slika 1.1 Prikaz generalne sheme GAN-a [2]

1.2. Generator model

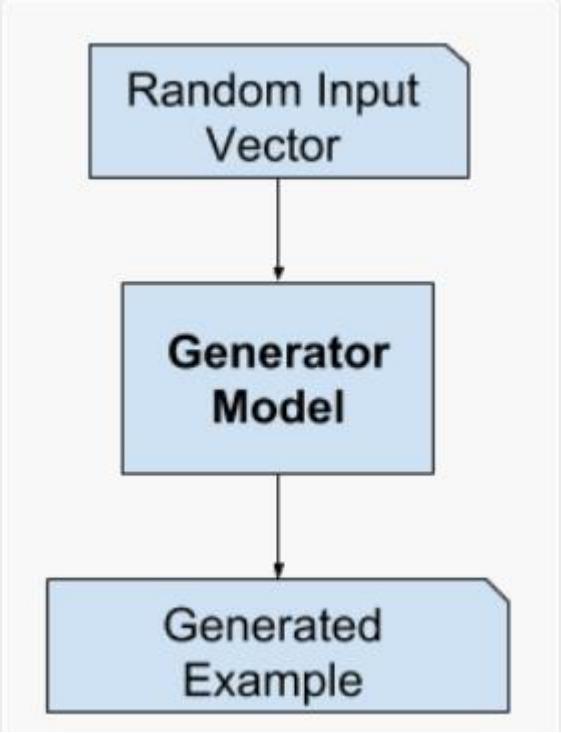
Generator model prima slučajni vektor fiksne duljine kao ulaz i koristi ga za generiranje uzoraka u domeni [1].

Vektor se izvlači iz Gaussove distribucije te služi kao početna točka generativnog procesa. Nakon obuke, točke u višedimenzionalnom vektorskom prostoru odgovaraju točkama u problemskoj domeni te tvore sažetu reprezentaciju distribucije podataka [1].

Ovaj vektorski prostor naziva se latentni prostor ili vektorski prostor latentnih varijabli. Latentne varijable, odnosno skrivene varijable, bitne su za problemsku domenu, ali nisu direktno promatrane. U slučaju GAN-a, generator model povezuje točke u latentnom prostoru s odgovarajućim primjerima u domeni, što omogućuje generiranje novih primjera izlaza korištenjem novih točaka u latentnom prostoru [1].

Nakon obuke, generator model koristi se za generiranje novih uzoraka [1].

Slika 1.2 prikazuje GAN Generator model.



Slika 1.2 Primjer GAN Generator modela [1]

1.3. Diskriminator model

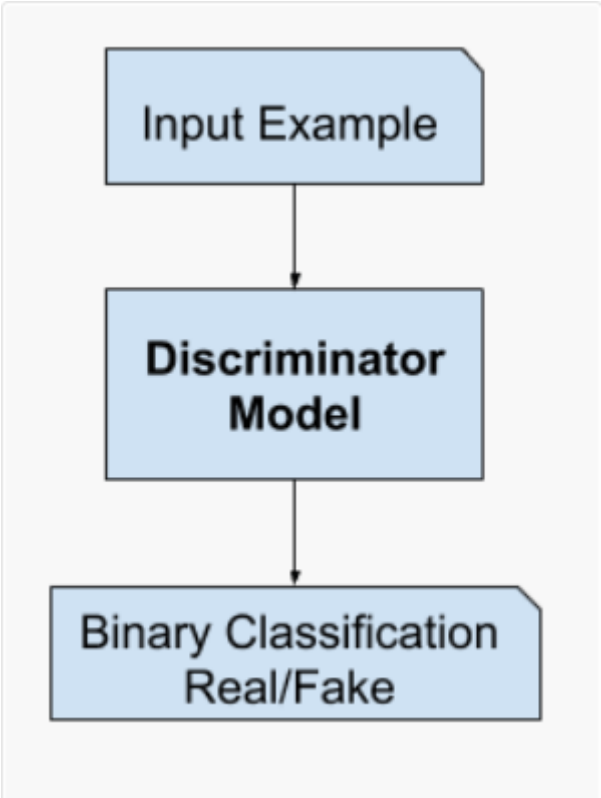
Diskriminator model prima primjer iz domene kao ulaz (stvarni ili generirani) te predviđa binarnu oznaku klase, stvarno ili lažno (generirano) [1].

Stvarni primjer dolazi iz skupa podataka za treniranje, dok se generirani primjeri izlučuju pomoću generator modela [1].

Diskriminator je, u suštini, običan model za klasifikaciju [1].

Nakon treniranja, diskriminator model se odbacuje jer nas zanima samo generator [1].

Slika 1.3 prikazuje GAN Diskriminator model.



Slika 1.3 Primjer GAN Diskriminator modela [1]

1.4. Arhitektura GAN-a

Dva modela, generator i diskriminator, treniraju se zajedno. Generator generira grupu primjera, a ti primjeri, zajedno s pravim primjerima iz domene, daju se diskriminatoru i klasificiraju se kao stvarni ili lažni

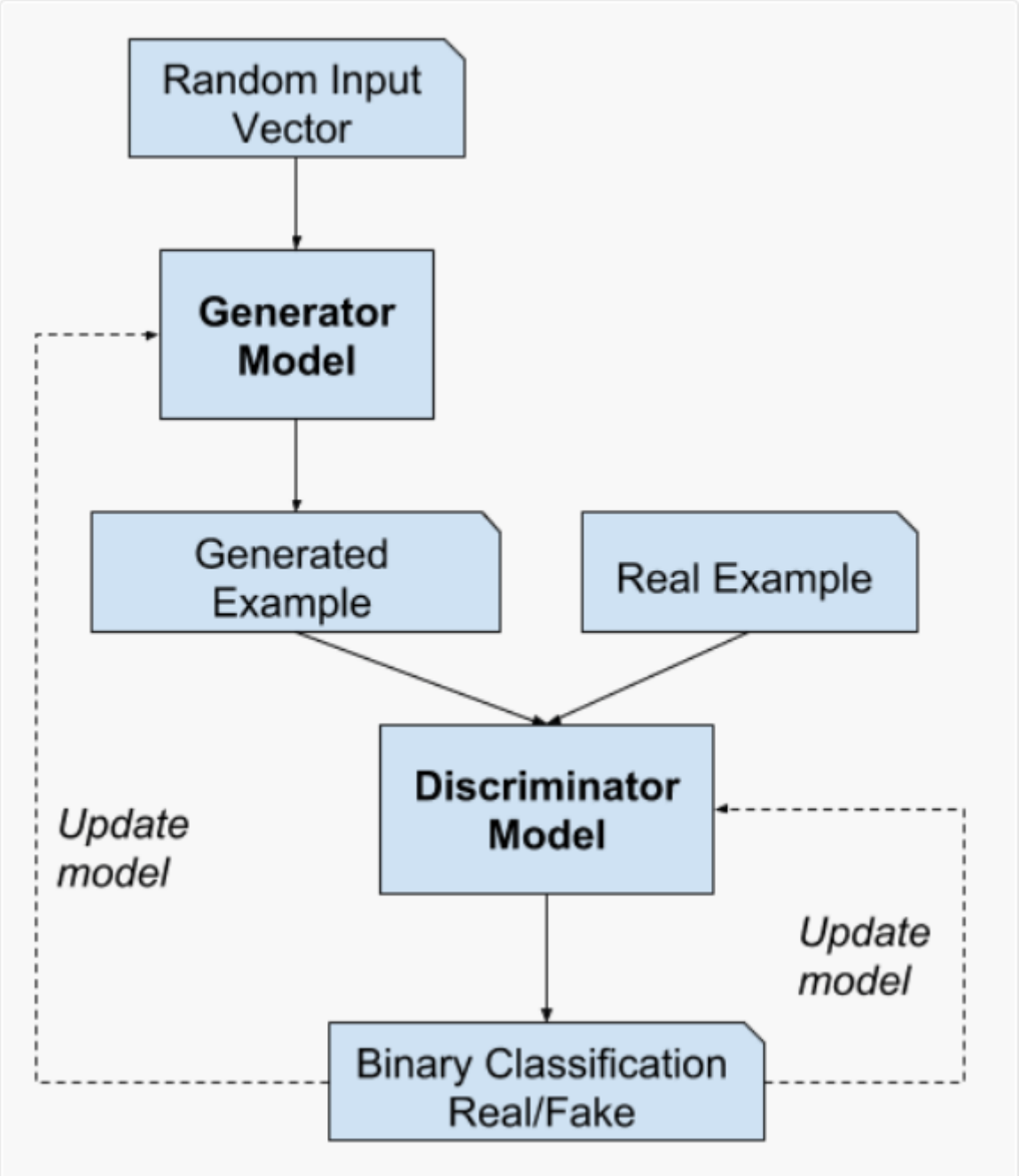
Diskriminator se potom ažurira kako bi postao bolji u razlikovanju stvarnih i lažnih primjera u sljedećoj rundi, a bitno je da se generator ažurira na temelju toga koliko su dobro, ili ne, generirani primjeri zavarali diskriminator [1].

Na ovaj način, dva modela se natječu jedan protiv drugog, suprotstavljaju se u smislu teorije igara i igraju igru nulte sume [1].

U ovom slučaju, nulta-suma znači da kada diskriminator uspješno prepozna stvarne i lažne primjere, nagrađuje se ili se ne vrši promjena parametara modela, dok se generator kažnjava velikim ažuriranjima parametara modela [1].

Kada generator zavara diskriminator, nagrađuje se ili se ne vrši promjena parametara modela, ali se diskriminator kažnjava i ažuriraju se parametri modela [1].

U idealnom slučaju, generator uvijek generira savršene replike iz ulazne domene, a diskriminator ne može primijetiti razliku i predviđa "nesiguran sam" (npr. 50% za stvarne i lažne primjere) u svakom slučaju. Ovo je samo primjer idealiziranog slučaja; ne moramo doći do ovog stupnja da bismo dobili koristan generator model [1]. Na slici 1.4 prikazana je arhitektura GAN-a.



Slika 1.5 Arhitektura GAN-a [1]

2. Implementacija i rezultati

U ovom poglavlju opisuje se skup podataka koji se koristi te kako je taj skup podijeljen. Također se opisuju osnove implementacije algoritma te dobiveni rezultati.

2.1. Skup podataka

MNIST skup podataka sastoji se od 70.000 slika rukom pisanih brojeva (0-9), od kojih je 60.000 slika namijenjeno za trening, a 10.000 slika za testiranje. Svaka slika je dimenzija 28x28 piksela i crno-bijela je [4].

2.2. Programska implementacija

Za implementaciju GAN-a koristit ćemo TensorFlow, popularnu open-source biblioteku koja nam omogućuje jednostavno definiranje, treniranje i evaluaciju modela [3].

Nakon učitavanja dataseta, slike se pretvaraju u dimenzije 28x28x1 i normaliziraju se vrijednosti piksela na raspon [-1, 1]. Korak normalizacije često se koristi u dubokom učenju kako bi se pomoglo u stabilnosti i konvergenciji treniranja.Nakon toga dolazi se do ključnog dijela implementacije, odnosno definiranja klase Generatora i Diskriminatora.

U našoj implementaciji GAN-a generator i diskriminator definiraju se pomoću Keras Sequential API-ja.

Model generatora sastoji se od nekoliko slojeva, uključujući potpuno povezane slojeve, slojeve normalizacije mini grupe (eng. *mini batch*), aktivacijske funkcije leaky ReLU i transponirane konvolucijske slojeve. Ulaz u generator je 100-dimenzionalni vektor slučajnih šumova, a izlaz je generirana slika s dimenzijama (28, 28, 1). Generator model dizajniran je da stvara sve složenije značajke kako se slojevi pomiču, počevši od ulaznog slučajnog šumova i završavajući s izlaznom generiranom slikom. Uporaba normalizacije mini-grupa pomaže stabilizirati treniranje i poboljšati kvalitetu generiranih slika.

Model diskriminatora sastoji se od nekoliko slojeva, uključujući konvolucijske slojeve, aktivacijske funkcije leaky ReLU, slojeve „ispadanja“ (eng. *dropout layer*) i potpuno povezane slojeve. Ulaz u diskriminator je slika s dimenzijama (28, 28, 1), a izlaz je jedna

skalarna vrijednost koja predstavlja vjerojatnost da je ulazna slika stvarna. Diskriminator model dizajniran je da nauči sve složenije značajke kako se slojevi pomiču, počevši od niskorazinske značajke kao što su bridovi i oblici te završavajući s visokorazinskim značajkama koje predstavljaju cjelokupnu strukturu slike. Uporaba dropout slojeva pomaže u sprječavanju pretreniranosti tijekom treniranja.

Funkcija gubitka diskriminatora koristi se za izračunavanje pogreške između predviđanja diskriminatora na stvarnim i lažnim slikama i koristi se za ažuriranje parametara diskriminatora tijekom treniranja. Funkcija prima dva parametra, real_pred i fake_pred, koji predstavljaju predviđanja diskriminatora na stvarnim i lažnim slikama. Izračunava se „binarni križni entropijski gubitak“ (eng. *binary cross-entropy loss*) između predviđanja diskriminatora i stvarnih oznaka (tj. 1 za stvarne slike i 0 za lažne slike). Gubitak diskriminatora izračunava se kao zbroj stvarnog gubitka i lažnog gubitka.

Funkcija gubitka generatora koristi se za izračunavanje pogreške između predviđanja diskriminatora na lažnim slikama i koristi se za ažuriranje parametara generatora tijekom treniranja. Funkcija prima jedan parametar, fake_pred, koji predstavlja predviđanja diskriminatora na lažnim slikama. Izračunava se „binarni križni entropijski gubitak“ (eng. *binary cross-entropy loss*) između predviđanja diskriminatora i stvarnih oznaka (tj. 1 za lažne slike). Ukratko, ova funkcija gubitka generatora pruža mjeru koliko dobro generator uspijeva prevariti diskriminatora da misli da su njegove generirane slike stvarne.

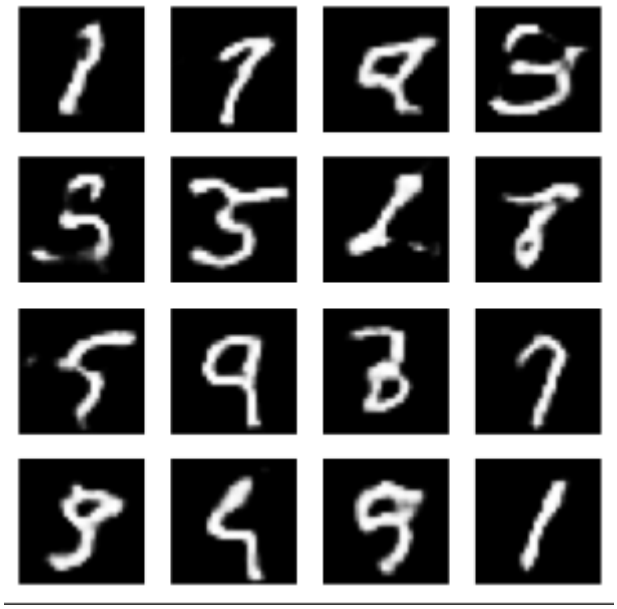
Također, definiraju se **dva Adam optimizatora** (jer imamo dvije mreže) za generator i diskriminator modele u GAN-u. Adam optimizator je popularni algoritam optimizacije za stohastički gradijentni spust koji se često koristi u dubokim modelima učenja. Prilagođava stopu učenja za svaki parametar temeljem prvog i drugog momenta gradijenta, što može pomoći u bržoj konvergenciji i izbjegavanju zaglavljivanja u lokalnim minimumima. U ovom slučaju, generator i diskriminator modeli se optimiziraju zasebno pomoću vlastitih objekata optimizatora.

Petlja za treniranje GAN-a (eng. *training loop*) provodi se u 50 epoha, gdje se u svakoj epohi ažuriraju modeli generatora i diskriminatora na temelju skupa stvarnih slika. Generira se skup od 16 lažnih slika u svakoj epohi pomoću generatora. Također se definira broj dimenzija za ulazni šum (noise_dim), što predstavlja broj dimenzija šuma koje će se koristiti za generiranje lažnih slika. Ovaj ulazni šum koristi se kao ulaz u generator tijekom generiranja lažnih slika.

2.3. Rezultati

Za potrebe prikaza rezultata pripremljene su generirane slike te gubitak diskriminatora i generatora tijekom epoha.

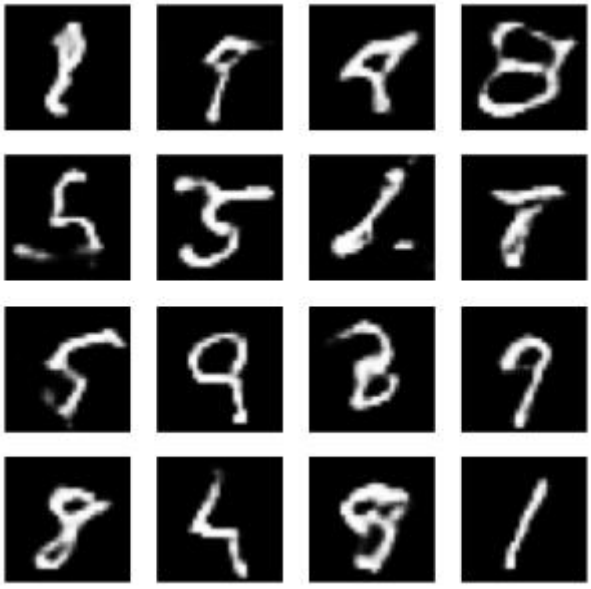
Slika 2.1 Epoha 1



Slika 2.2 Epoha 20

Slika 2.3 Epoha 30

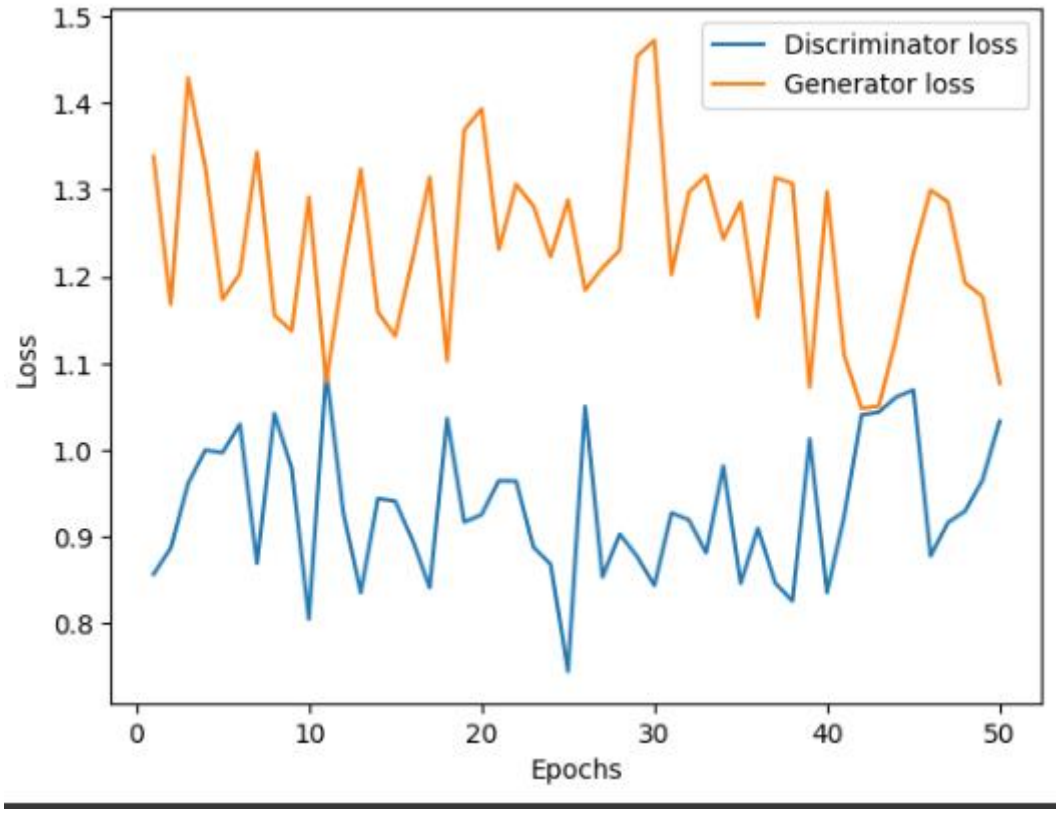
Slika 2.4 Epoha 40



Slika 2.5 Epoha 50

Na slikama 2.2 – 2.5 možemo golim okom vidjeti kako generirane slike znamenaka s porastom epoha sve bolje i bolje imitiraju stvarne slike znamenaka.

Isto tako, na slici 2.6 vidljivo je kako s rastom epoha gubitak diskriminatora lagano raste dok gubitak generatora opada. Objašnjenje iza takvih grafova je takvo da generator pokušava smanjiti udaljenost između generiranih podataka i stvarnih podataka, dok diskriminator pokušava maksimizirati svoju sposobnost razlikovanja između generiranih podataka i stvarnih podataka.



Slika 2.6 Gubitak generatora i diskriminatora tijekom epoha

Zaključak

U ovom seminaru bavili smo se GAN-om, vrstom neuronskih mreža koja omogućuje generiranje novih primjera iz skupa podataka. Implementirali smo vlastiti generator i diskriminator model te postigli zadovoljavajuće rezultate u generiranju novih slika.

Važno je naglasiti da GAN-ovi nisu korisni samo za generiranje slika. Oni se mogu koristiti i za generiranje zvuka, glazbe, teksta i drugih vrsta podataka. GAN-ovi su jedan od najpopularnijih i najuspješnijih pristupa u generiranju podataka. Međutim, kao i kod svih metoda strojnog učenja, ključni su podaci za učenje, pa je bitno imati dobar skup podataka kako bi se postigli dobri rezultati.

Sa sigurnošću možemo reći da će GAN-ovi s daljnjim razvojem tehnologije i algoritama te gomilanjem podataka postati još učinkovitiji u generiranju realističnih primjera podataka.

Literatura

[1] *A Gentle Introduction to Generative Adversarial Networks (GANs)*, Poveznica: https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/

[2] *Generative Adversarial Network*, Poveznica: https://www.geeksforgeeks.org/generative-adversarial-network-gan/

[3] *Tensorflow*, Poveznica: https://www.tensorflow.org/api_docs/python/tf

[4] *MNIST digits classification dataset*, Poveznica: https://keras.io/api/datasets/mnist/

[5] *How to build a GAN*: https://medium.com/@simple.schwarz/how-to-build-a-gan-for-generating-mnist-digits-in-pytorch-b9bf71269da8

[6] *How to develop a GAN from scratch*, Poveznica: https://machinelearningmastery.com/how-to-develop-a-generative-adversarial-network-for-an-mnist-handwritten-digits-from-scratch-in-keras/

Sažetak

Ovaj rad fokusira se na implementaciju Generativne suparničke mreže (GAN) korištenjem TensorFlow okvira za generiranje MNIST brojeva. GAN-ovi su vrsta neuronske mreže koja može naučiti generirati podatke tako što suprotstavlja dva modela: generator i diskriminator. Proces implementacije uključivao je treniranje GAN-a na MNIST skupu podataka, koji sadrži 60 000 slika rukom pisanih brojeva od 0 do 9 u sivoj razini. Generator je treniran za generiranje realističnih slika brojeva, dok je diskriminator treniran da razlikuje između stvarnih i lažnih brojeva. U radu se istražuju razni aspekti GAN arhitekture, uključujući funkciju gubitka, tehnike treniranja i hiperparametre. Rezultati pokazuju da je GAN sposoban generirati realistične slike brojeva.

Ključne riječi: generativne suparničke mreže, GAN, generator, diskriminator, MNIST

Summary

This work focused on the implementation of a Generative Adversarial Network (GAN) using the TensorFlow framework for generating MNIST digits. GANs are a type of neural network that can learn to generate data by pitting two models against each other: a generator and a discriminator.

The implementation process involved training the GAN on the MNIST dataset, which contains 60,000 grayscale images of handwritten digits from 0 to 9. The generator was trained to generate realistic-looking digits, while the discriminator was trained to distinguish between real and fake digits. The thesis explored various aspects of the GAN architecture, including the loss function, training techniques, and hyperparameters.

The results showed that the GAN was able to generate realistic-looking digits.

Keywords: generative adversarial networks, GAN, generator, discriminator, MNIST

Skraćenice

GAN	*Generative adversarial network*	generativne suparničke mreže
eng.	*English*	engleski
ML	*Machine learning*	strojno učenje
AI	*Artificial intelligence*	umjetna inteligencija